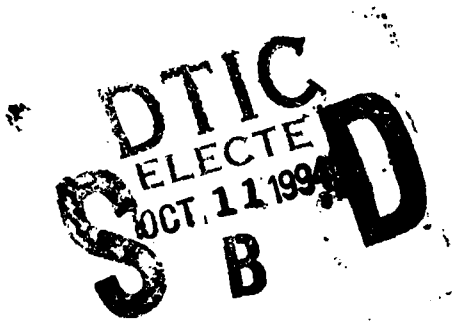
NRL/FR/5520--94-9733

An Integrated Knowledge Acquisition and Database Management System

C. B. Barclay
J. A. Molnar

Communication Systems Branch
Information Technology Division

August 26, 1994



Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 26, 1994	

4. TITLE AND SUBTITLE

An Integrated Knowledge Acquisition and Database Management System

5. FUNDING NUMBERS

DN 580-095

6. AUTHOR(S)

C.B. Barclay and J.A. Molnar

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/5520--94-9733

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Sea Systems Command
Washington, DC 20362-5101

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

FISDM is a menu driven database interface that was developed to enhance the acquisition and management of knowledge acquired for Fault Isolation System Knowledge Bases. This software was of general purpose and could be accessed on any ASCII terminal. FISDMPro enhanced the basic FISDM software by providing a Graphical User Interface (GUI) and Expert System to aid knowledge base creation. FISDMPro was developed with the GUI development tool TAE (Transportable Application Engine). The Expert system portion of FISDMPro uses the C Language Integrated Production System (CLIPS) expert system engine, and rules were developed that govern the creation of the major components of a FIS knowledge base. The Interface now presents a graphical method of creating the knowledge base by simply defining modules as blocks and providing interconnections between the blocks. The interconnections are governed by the physical interrelations of the modules, and the expert system generates rules relating to the interconnections. A presentation of the capabilities is shown along with the discussion of the software dynamics.

14. SUBJECT TERMS

Fault isolation
Database
Knowledge acquisition
Graphical interface
Expert system

15. NUMBER OF PAGES

21

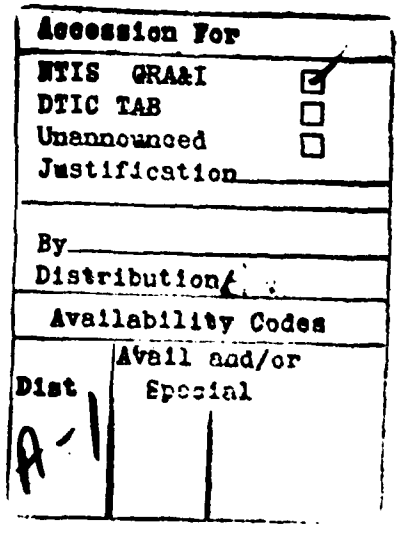
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS



AN INTEGRATED KNOWLEDGE ACQUISITION AND DATABASE MANAGEMENT SYSTEM

INTRODUCTION

The Fault Isolation System (FIS) is a diagnostic reasoning system used to detect and isolate faults in electronic systems using analog circuitry. The FIS is an expert system shell developed by the Navy Center for Applied Research in Artificial Intelligence (NCARAI)[1]. The FIS uses knowledge databases that describe an electronic system as a model of causal relationships. Although the FIS effectively addresses knowledge representation, knowledge acquisition from schematic diagrams of these electronic systems reduces the FIS's usability [2].

The development of the FIS created a need for data, in the form of knowledge bases, to assist in fault diagnosing units under test (UUT). These data were retrieved from schematic diagrams and involved modules, test points, and connections. Data were read from the diagram by an expert, the knowledge engineer, who had to keep track of incoming and outgoing connections to each module. In the absence of proper tools, this task was both tedious and error prone. It was thought that by automating the task, the knowledge engineer would be free to examine the final knowledge base, and less skilled workers could enter the data. Productivity gains would be realized as knowledge databases were created in less time with a higher degree of correctness.

The first attempt at creating a FIS tool was an ASCII based Database Manager [3]. Although this was an improvement over entering data through the FIS editor, determining what data were important still required an expert. In addition, the interface was less user friendly than desired. It was decided that a graphical user interface would allow additional functionality to be added, with little additional learning curve. X-Windows, a graphical interface, was chosen over other graphical interfaces because of its portability. X-Windows ensures consistency among different computer platforms. No matter what platform the user chooses to execute the software, the look and feel are the same. Moreover, if FISDMPro is running on a server, all machines connected to that server that support an X-Windows environment can access the software.

Knowledge databases are dynamic entities because of hardware updates and modifications. A need was seen for simpler addition and modification of data. A database format was selected for this task using Empress database software [3]. The database format allows rapid viewing and updating of data in a manner familiar to users of commercial databases. Inclusion of a database format allows knowledge engineers an alternate method to modify data and provides for an integrated knowledge acquisition system.

Additional functionality was built into the software that allows easier knowledge acquisition. According to Eisenstadt et al., "...knowledge engineers have not had access to visual programming tools [to] assist them during the critical early phases of knowledge acquisition." [4] We believed that not only was knowledge acquisition a problem with FIS, but the need for better tools was inherent to all expert

Manuscript approved December 9, 1993

systems. The system designed uses FIS data models, but through the use of the CLIPS expert system, other data types may be created without modification or recompilation of the main program.

The basic idea behind the software is a simplified rendering of the objects the knowledge engineer works with. The knowledge engineer is given a blank area in which to copy data seen on the UUT diagram. The engineer may draw modules and their connections and let the automated routines generate the proper datafiles.

CLIPS

CLIPS is the C Language Integrated Production System developed by NASA [5]. It is easily linkable to C programs; this was a requirement of any software we used. New functionality may also be added to CLIPS, which increases its usability with complex data types. We thought that by using CLIPS, a model could be developed that allows a number of different knowledge databases to be created, each using a different CLIPS program. Since the CLIPS file does not have to be compiled with the program, it is easy to change the knowledge base format to accommodate the needs of other expert systems. The CLIPS program takes the knowledge acquired on screen and transforms it into the appropriate output. In this program, appropriate output is defined as a file that is accepted by the FIS compiler. Data comes in a certain prespecified format consisting of module name and links. These data may be manipulated by using CLIPS rules to produce the desired output.

CLIPS receives data information from the screen in the form of a data structure that details the connection information. This knowledge fires rules that create new knowledge. The rules make the software flexible and powerful because a redefinition of a rule can change the output format drastically. Any expert system that requires data and shares information through connections would be able to be represented in this manner through a redefinition of the rules. Since this is a common feature of expert systems that defines the cause-effect relationship of physical systems in the form of abstract data models, generally this technique should be applicable to many knowledge base construction problems. Similarly, many heuristic or experiential expert systems employ rules concerning causal relationships, if-then rules; such knowledge could be represented using our system.

As Swaby states, "Although we could have produced the knowledge bases manually, we wrote a program to build them automatically [because] the output format might be changed...however, the knowledge base builder need be modified only once to rebuild a whole knowledge base incorporating the changes." [6] His idea is similar to ours, but we take this one step further: what if there is not a small change in the knowledge base, but a drastic one, i.e., a different knowledge base altogether? The same mechanism should be able to handle both problems. The modular nature of our system allows the simple modification of rule structures to process any knowledge format with minimal additional effort required to produce the correct interface for user interactions.

SOFTWARE FUNCTIONALITY

Many of the FISDMPro features are identical to the features of the original FISDM [2]. The interested reader should refer to that document for detailed information on many of the functions. This report stresses the improvements made to FISDM and details the enhancement to the user interface and the embedded expert system features. The following text presents these features in detail.

Main Menu

Menu selections in FISDMPro are made by pointing and clicking, as opposed to keyboard entry that was required in FISDM. A user selects an option by placing the mouse pointer on a radio button and depressing the left mouse button. Once the desired button is visibly in, move the mouse pointer to the *OK*

button and depress the left mouse button. Notice the sunken area around the *OK* button. This indicates the default push button on the panel. To activate the default button, press the Return key on the keyboard with the mouse pointer anywhere in the window.

A *HELP* button has been added to the selection panel to provide information on the features of the panel. When this button is depressed, the mouse pointer is replaced by a question mark. Point the question mark at the item for which help is needed and depress the left mouse button. Another window will appear with information about the item in question.

The radio buttons detail the different choices available from the program. Each, when chosen, will remove the current window from the screen and replace it by the desired window. In most cases the radio buttons correspond directly to the menu choices of FISDM.

A *Delete a Database* option has been added to the menu. The *Delete a Database* option in the Main Menu lists all databases available in FISDMPro. The user may highlight one at a time for deletion. The databases are unconditionally deleted; there is no way to undo this function.

Expert Knowledge Acquisition

Knowledge acquisition is a time consuming task during the process of defining an expert system. Some methods have been designed to automatically generate knowledge bases, but these methods typically rely on previously entered data. The goal of this part of the project was to demonstrate an easier method for acquiring knowledge and to present an integrated system to manage these data.

Since the FIS is a model based expert system shell, modeling systems of any complexity can lead to quite large knowledge bases. The knowledge engineer must extract the data for the knowledge base from schematic diagrams of system components. Entering these data through text entry systems can be a tedious and error prone task. To simplify the task, it is necessary to remove the text-based input and rely more upon the interrelationships expressed on the schematic. Schematic diagrams are drawings that relate modules and the connections between the modules. Some module interconnections have been designated test points by the designer. These connections are treated differently by the FIS because they express a point where a technician can perform measurements on the system. The FIS knowledge base must possess the knowledge differently in order to express these testpoints to the technician during a fault isolation session. The graphical interface that we developed can represent these modules, their interconnections, and the testable interconnections in a manner that is similar to the graphical presentation of the schematics, and thereby is more intuitive than plain text. In addition, it is more desirable to be able to view and modify records in a database style format. Last, it is necessary to output data in the FIS readable format. All three capabilities are common to design engineers. Current software for the design of electrical and mechanical devices often integrates graphical capture capabilities with database storage methods, and the final design must often be produced in a separate electronic format for automated prototyping. We have borrowed from this methodology and would hope that future work could extract the FIS knowledge directly in the design phase. Until then, we have created a similar design tool for the knowledge engineer.

Knowledge acquisition is accomplished by using a graphical area for data input. The knowledge engineer draws modules on the screen that represent the modules in the schematic. The modules are connected with lines that capture the interrelationships shown on the schematics. The connection process is accomplished by clicking first on the module where the connection originates and clicking a second time on the module where the connection ends. These connections are the basis for the FIS data file. There are two parts of the FIS knowledge base that are created in this manner: the rules and the tests. Tests are a subset of the rules in the sense that connections are expressed as rules and any connection that is testable becomes

a test for the FIS knowledge base. The rules section of the FIS knowledge data file is formulated according to the following criteria:

- An input to a module is connected to every output from that module, unless the knowledge engineer states otherwise.
- For every connection between an input to the module and an output from the module, there is a rule with that input as the cause of failure.
- For every output from a module, there is a rule with that module as the cause of failure.
- An input's cause must be the same as the output's effect in the upstream module.
- For every terminal and parameter used in the causal rules, there must be a correlation to the terminal and parameter used in the test information file.
- If there is a "J" in a parameter name, then it is a test point.

An extended discussion of knowledge base formulation, with examples, is found in the report by Molnar [2].

The first rule pertains to how incoming connections to a module are connected to outgoing connections. By default, all incoming connections are connected to all outgoing connections. This can yield huge knowledge bases if there are many inputs and outputs in a module. In general, the knowledge engineer limits these interconnections to those defined on the schematic. The next two rules deal with how failures are propagated through the FIS system. To construct a reasonable model of the UUT, the FIS must be told that if a particular module fails, all the downstream modules should turn up bad.

The above rules were written in the CLIPS expert system and provide a method of generating a valid FIS data file. Facts are asserted onto the CLIPS stack through C functions. The facts contain data such as module name, connections from the module, and other data pertaining to the modules. CLIPS pattern matches in a forward chaining environment to develop the rule and test lists.

When *Expert Knowledge Acquisition* is chosen from FISDMPro, a large window is created covering most of the screens, as shown in Fig. 1. There are options on the left side of the window, and most of the window taken by X workspace. This X workspace is where data are represented. The knowledge engineer may create data, or may input data from a file or the database manager. To create data, first the modules must be created.

Modules are created by selecting *Add Module* from the options list and entering a module name in the module name data input area located in the top left corner of the drawing board. Position the mouse pointer on the X workspace where the upper left corner of the module is to be located. Press the left mouse button and keep it down. A small square will remain where the original depression occurred. As the mouse is moved toward the bottom-right, the square will enlarge while the left mouse button is depressed. Once the size of the area is sufficient, the mouse button is released. A rectangle will be drawn over the selected area. Modules should not be constructed on top of one another as there will be no way to access the "hidden" module.

When two or more modules are built, connections may be defined. Select the *Define Connections* option and move the mouse pointer to the module that will output the data. Click the mouse button once; as a result the module color changes to green, which is the lighter colored module in Fig. 1. Move the mouse pointer to the module that will input the data from the first module. Once again click the mouse button once; this changes the module color to red, which is the darker colored module in Fig. 1. The window shown in the figure will pop up requesting information about the connection. To generate a successful FIS file, appropriate information must be entered. A link name is required to establish a reference for accessing the connections during the creation of the knowledge base. Usually the link name would be the name of a physical terminal that exists on the module [2]. Each connection must have a distinct link name. The

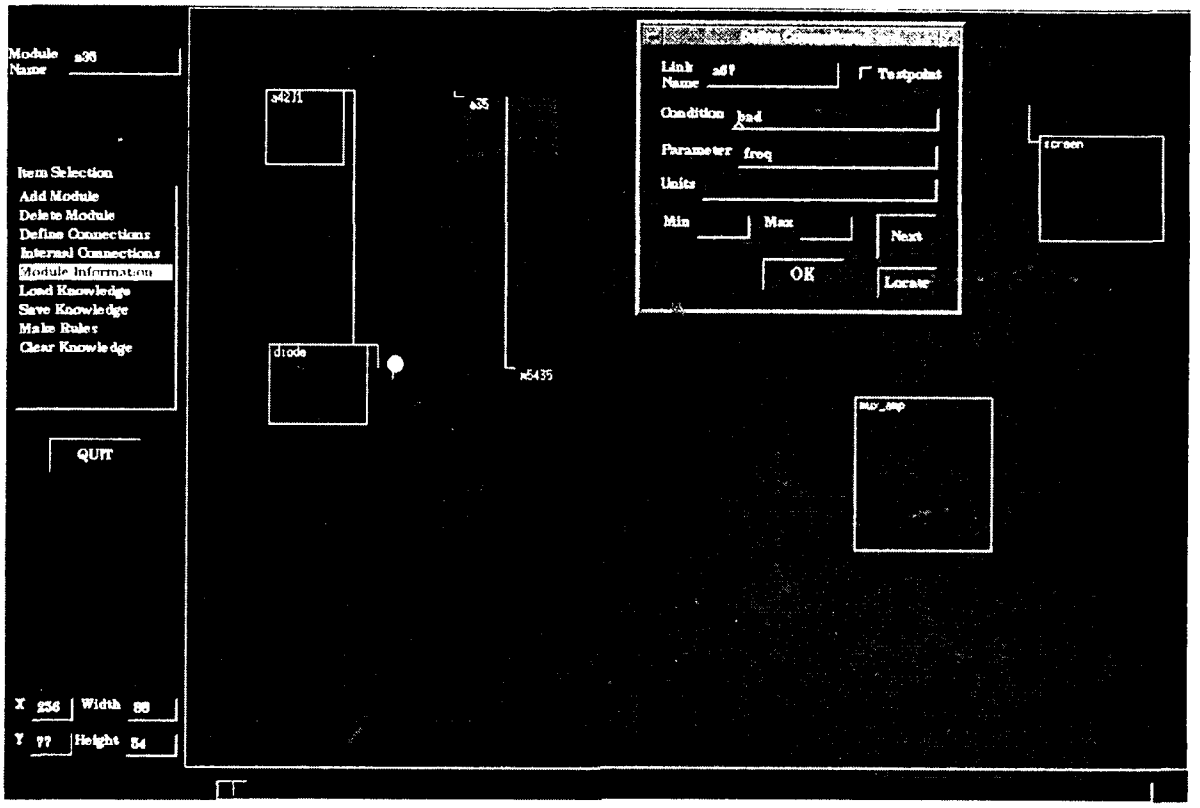


Fig.1 - FISDMPro Knowledge Acquisition Drawing Board for graphically defining modules and their interconnections

parametric measurement that would exist between the modules. Each connection must have information entered in these fields to establish a proper connection (and subsequent rule). Additionally, if the *Testpoint* box is checked, then that indicates to FISDMPro that this connection is a testpoint and that a parametric test can be performed at this connection by a technician. Subsequently, testpoint information must also be established in the *Define Connection* window. This is the information that is supplied to the *Units*, *Min*, and *Max* boxes. When finished, pressing the *OK* button creates the connection. For information about previously created connections, select *Module Information* from the selection list and click on the module. Information will be displayed about that module's outgoing connections. This window is the same as the window used to input information about module connections, but it has the additional buttons *Next* and *Locate*. The *Next* button moves to the next outgoing connection until there are no more connections to be displayed. The *Locate* button finds what module the connection is connected to by turning the current module green and the connected module red. In the event the module is connected to itself, a message is displayed. On large knowledge bases it is sometimes necessary to use the drawing area scrollbar to find the location of the connection on the drawing.

Note that the *Define Connections* command defines only the connections between modules. In addition, the internal connections of the module must be defined. The FISDMPro default for defining internal connections is to provide a configuration for a module that has all inputs connected to all outputs. To override this default setting, select *Internal Connections* from the options menu. Position the mouse pointer over the module to be modified and click the left mouse button. A window will appear displaying three menus. The left menu contains the names of all incoming connections. The center menu contains the names of all outgoing connections. Select one name from the left menu (it will highlight when selected). Select all outputs this will be connected to from the center menu. The selected items will be duplicated onto

the right menu. When finished with each input, press *OK*. It is necessary to repeat this process for each input, otherwise the undefined input connections are connected to all output connections. While overspecifying the internal connections does not result in an illegal format for the FIS knowledge base, it acts to improperly model the physical situation. Subsequently, the FIS will react more slowly to indications of faults obtained from measurement information; FIS will have to perform more computations; the ambiguity set generated by the FIS will remain large for a longer period during the fault isolation process; and finally, FIS may produce inaccurate results. FISDMPro users are **strongly cautioned** to verify the internal connections on all modules to avoid unexpected FIS behavior.

Facile creation of FIS files is performed by using the embedded CLIPS engine. Selecting *Make Rules* from the menu enables the conversion of graphical data on the screen to the FIS *.v* file format. Module information such as connections, causes and effects, is loaded onto the CLIPS stack. The connection information provides the foundation for the creation of the testlist and the rulelist sections of the FIS data file. This does not completely specify the contents of the *.v* file; other portions, such as the definition of precondition and order sections, could not be easily automated.

Loading of knowledge is achieved through the use of another CLIPS file. The method involved in this conversion is as follows: if the effect from one module is the same as the cause for another module, the two modules are connected. The effect from the second module tells of the internal connections. The cause of the first module should be the same as the module name. This means that only the outputs will be placed on the stack. Loading of knowledge in this manner is done by pressing the *Empress* push-button on the *Load Files* dialog window, as shown in Fig. 2. Pressing the *Unix* push-button gives a listing of files available in the Unix file system.

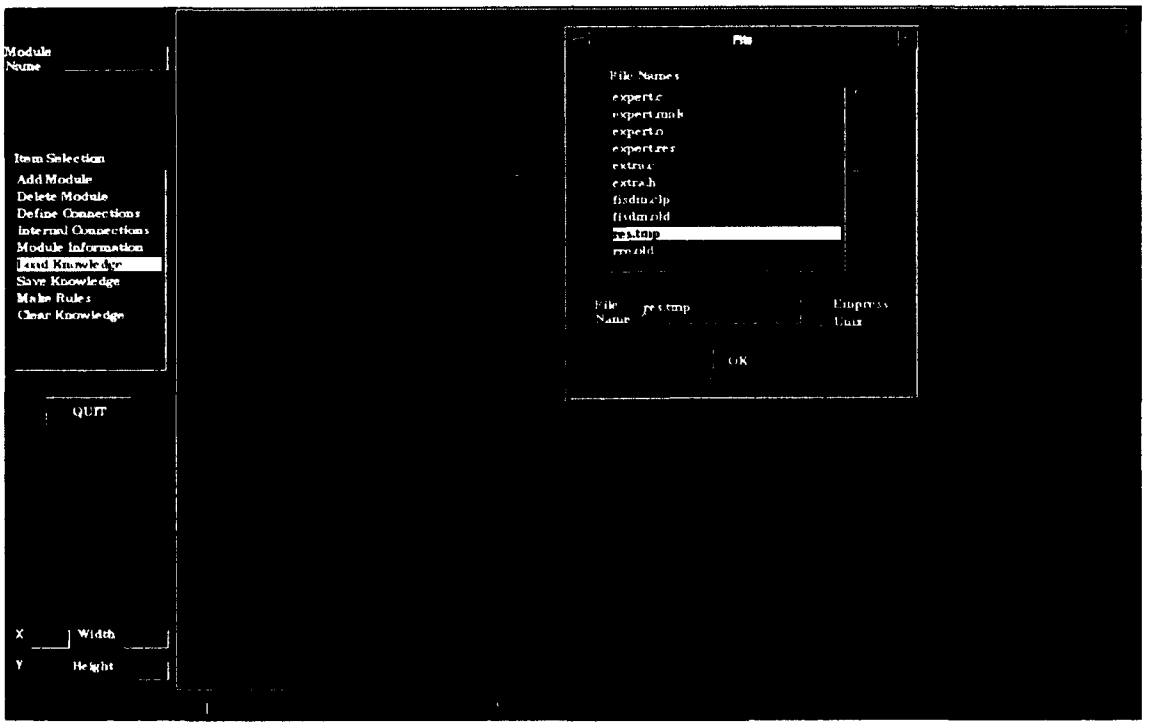


Fig. 2 - Display window for loading files into FISDMPro

The *Save Knowledge* option provides fast storage of the graphical knowledge without conversions. Data files created in this manner save all module and connection information except for internal connection

information. The only method to save internal connection information is to make a FIS file. This option cannot save data to the Empress file format directly.

Conversion Menu

FISDMPro accepts three data formats. Only two are in common use; they are comma delimited and FIS formats. The third is a text format common to many spreadsheets (the data is grouped into columns.) The same selection rules are followed on this menu as all others. The conversions are divided in two groups: conversion from FISDMPro to another format or conversion from another format to FISDMPro. In each case the file input window, shown in Fig. 3, behaves somewhat differently.

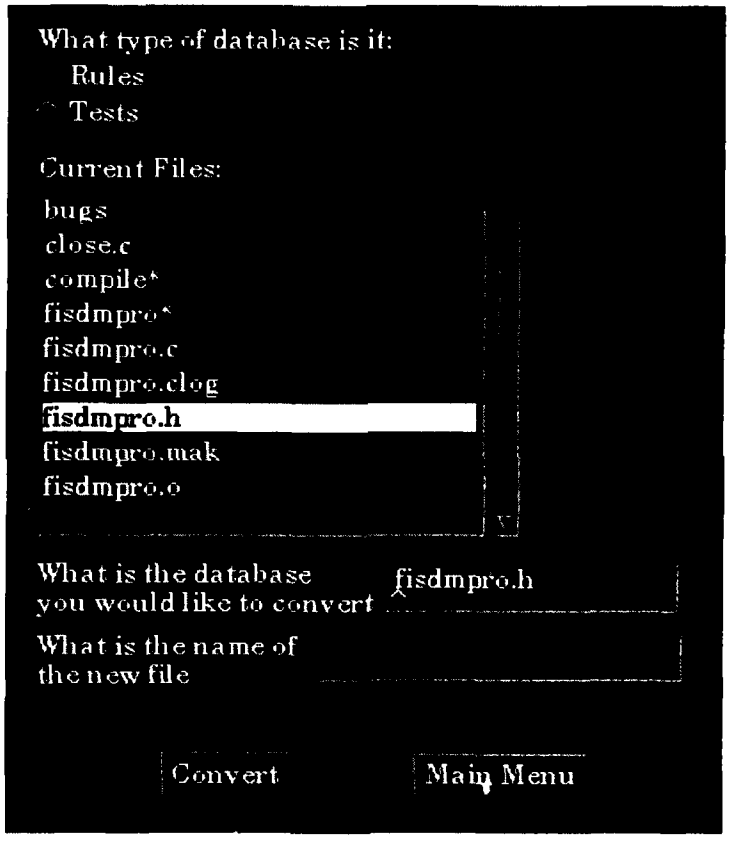


Fig. 3 - File input window for conversions

When inputting an outside file into FISDMPro, the user is presented with the window shown in Fig. 3. A list is displayed of files in the current directory. There are two ways to input files from directories other than the current directory. If the exact location of the file is known, it may be typed into the space labeled *What is the database you would like to convert.* The second method is to use the mouse on the directory window. By clicking once on the ../ symbol at the top of the directory, the directory is moved one level up. Inversely, by clicking on a directory name, the current directory becomes that directory. When the proper file is seen, the user may either click on the file name, causing it to appear in the first box, or the user may type the name in the first box by placing the mouse arrow in the box. The last step is to give a name to the new file. If a file with the same name exists in the database, the user will be prompted for a nonconflicting name.

When outputting a file into the comma delimited format, the user is presented with a window similar to that in Fig. 3, except that the database to be converted is selected from the FISDMPro database listing

and converted to a file in the current directory. Converting to and from FIS editor format presents a window similar to that in Fig. 3, but the radio button choices of *Rules* or *Tests* are not listed. Instead, the user must specify these files. The FIS editor format requires that both rule and test files be present in any database. Notice there is no capability to output in the text format by using the conversion menu. The text format is no longer supported; the only way to output in a format similar to text is to choose the *Print a Database* option from the Main Menu. This is only an approximate conversion and merely included for completeness and to provide a convenient method for displaying the data.

View Menu

Often, a user may wish to examine the rule or test components of the knowledge database. The Main Menu option *Work with a Database* allows the knowledge engineer not only all the functionality of the FIS editor, but provides many of the commands associated with modern databases. Figure 4 shows the selection window for this menu.

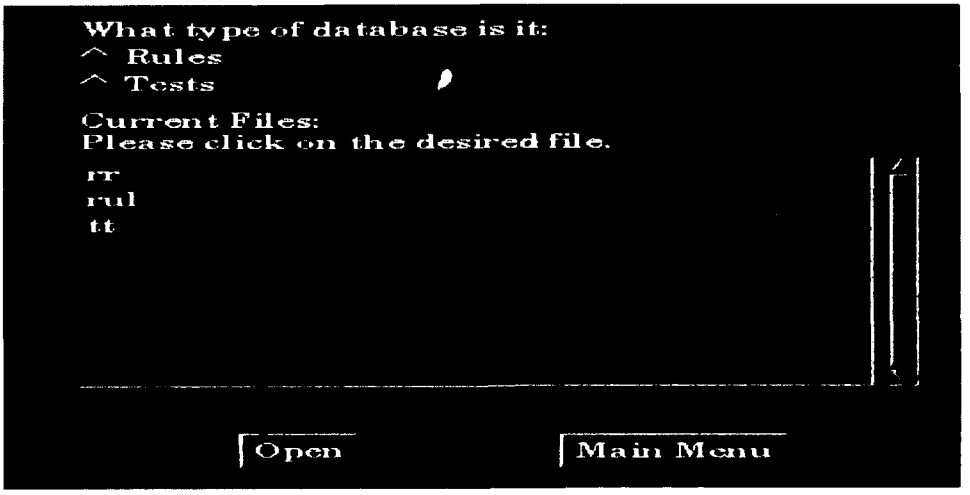


Fig. 4 - File selection window for working with an existing database

The window provides names of currently defined databases. Move the mouse to one of the names and click the left mouse button to highlight the desired name. Click the left mouse button on either *Rules* or *Tests*, depending on the database. Finally, click the left mouse button on *Open* to open the selected database. Only one database may be open at any one time. Figure 5 shows a new menu that presents options for database type manipulation.

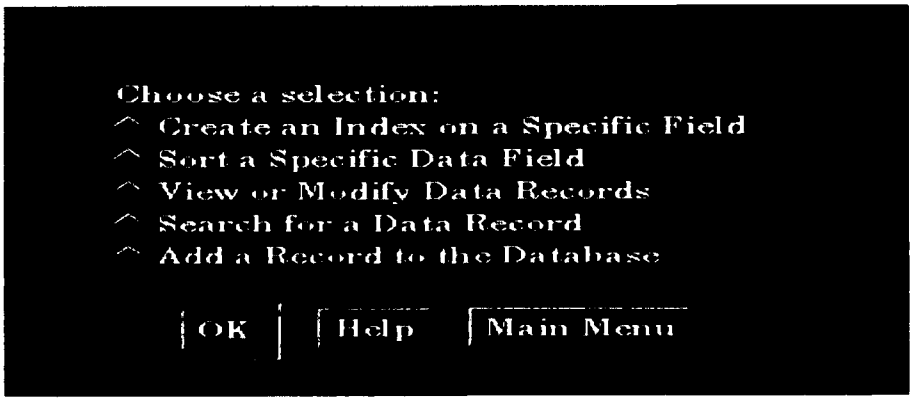


Fig. 5 - Database manipulation option menu window

The user may choose to create an index to improve sorting and searching individual fields in large data files. This is done by pressing the radio button corresponding to the desired field to be indexed. Since these indices take up extensive memory resources, the user is alerted to all current indexes when this option is selected. Figure 6 is presented to the user as an indication of currently indexed fields. The user has two options: click the left mouse button on the *Proceed* button, which allows the creation of a new index, or click the mouse on an index to highlight it and press *Delete*. This deletes only the highlighted indices. All indices are deleted when the program is exited.

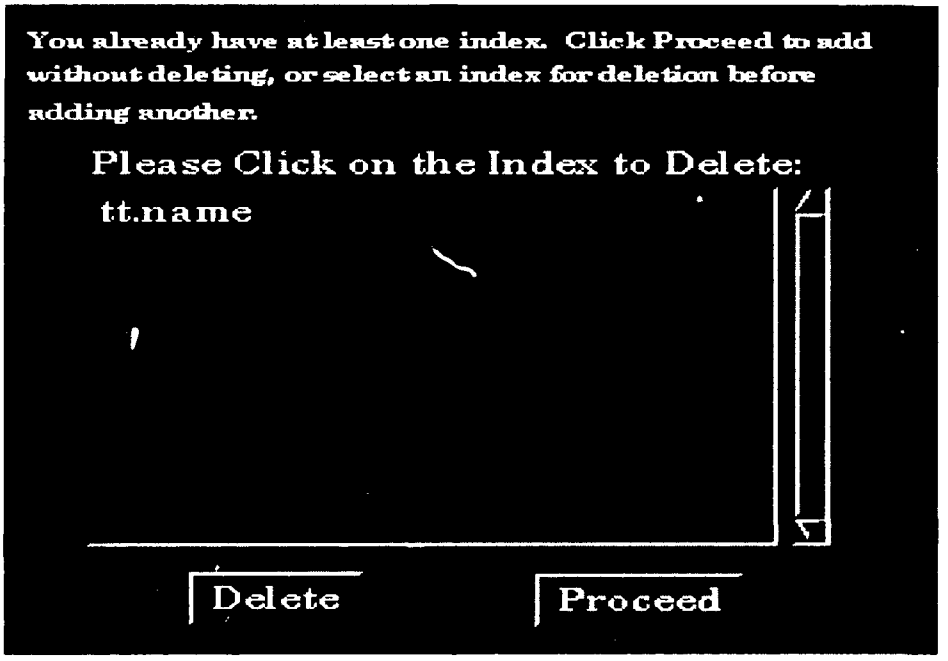


Fig. 6 - Warning window that indicates currently existing data indices

FISDMPro has the capability to sort fields in the database. When the option to sort fields is selected, the window, shown in Fig. 7, appears to the user allowing the user to select data fields for sorting. This window is slightly different depending upon whether a rule or test database is being examined, since different fields exist in each database type. Once the field is selected, the user may select the order of the sort, and where the output should be directed. The order of the sort specifies whether the sort is ascending (A-Z) or descending (Z-A). The output may be either directed to a new window on the screen or to a file specified by the user. When the *Screen* radio button is pressed, the file name input area disappears from the window. The *Begin Sort* button is depressed to start the sort procedure.

The greatest advantage of FISDMPro over the FIS editor is the ability to manipulate the knowledge in the data records. This is accomplished through the *View or Modify Data Records* option. Selection of this option presents the user with the window shown in Fig. 8 (which is shown for a test database), where each of the fields is identified and the value of a single record in the database is shown. The first record in the database is always shown when this option is selected. The user has five options for the next command; *Next* is the default condition. To modify the current record, move the mouse arrow into the data field to be modified and click. A carrot symbol should be visible in the record. Once modifications are finished on the record, press the *Update* radio button and *OK*. The user may then select another command. If the record is updated and the *Update* button is not selected as described above, the modifications will not be permanent. The user may also highlight parts of the data for deletion or copy. To unconditionally delete the current record, select the *Delete* radio button and *OK*. This window is unique among the windows in this program

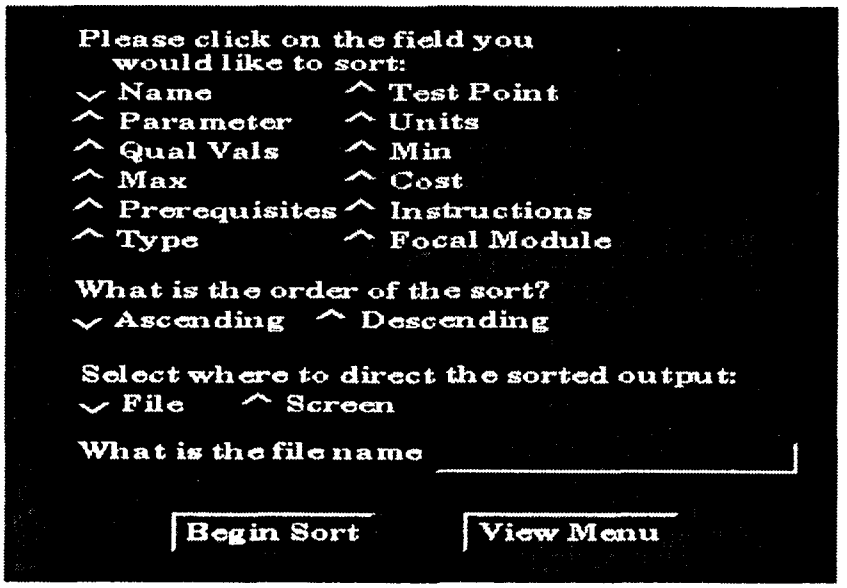


Fig. 7 - Sort selection window for a test database

— it may be resized if the data in the fields are too long to be fully displayed on the default size window. To do this, move the mouse to a corner; the cursor will change from an arrow to two sides of a square when it is in the proper position. Hold the left mouse button down and move the mouse to the desired shape. By increasing the horizontal length, all data records should fit in the window.

The final option available from the database type manipulation menu is the *Skip* command. This command creates another window that lays on top of the existing data record. There are four skip options: skip ahead a certain number of records, a positive integer; skip to a particular test or rule name; skip to the top of the database; or skip to the bottom of the database. When typing a name that has to be found, it is not necessary to type the full name. The search will skip to the first occurrence in the database of the letters provided.

The *Search* option from the View Menu is similar to the search used in the skip above. Figure 9 displays the search window presented to the user for a test database. The user needs only select the field type that is to be searched and enter part of a word in the data entry area. The program will find all matches for the text string of the indicated field. These matches are listed in a separate window that is controlled by a scroll bar.

The last option available on the View Menu is *Add a Record to the Database*. The format of this data input is similar to that of the *Create a Database* command in the Main Menu; both will be discussed below. From the Main Menu select, the user must give a name to the database and state whether it is rules or tests.

Adding records to a database uses a form similar to the viewing format, shown in Fig. 8, except that default values are present in some of the fields while other fields are empty. If the default values are acceptable, do not change them. Enter the appropriate data into the blank fields. Some of the fields check to confirm that the data is entered in the correct form. If the data entered does not conform with the template, the program will give an error message and erase the incorrect data. As was explained in the previous paragraphs, it is necessary to have the mouse pointing at the object that is to be modified. These fields conform to standard X-Window protocol. If the user highlights data in a field, press the middle mouse

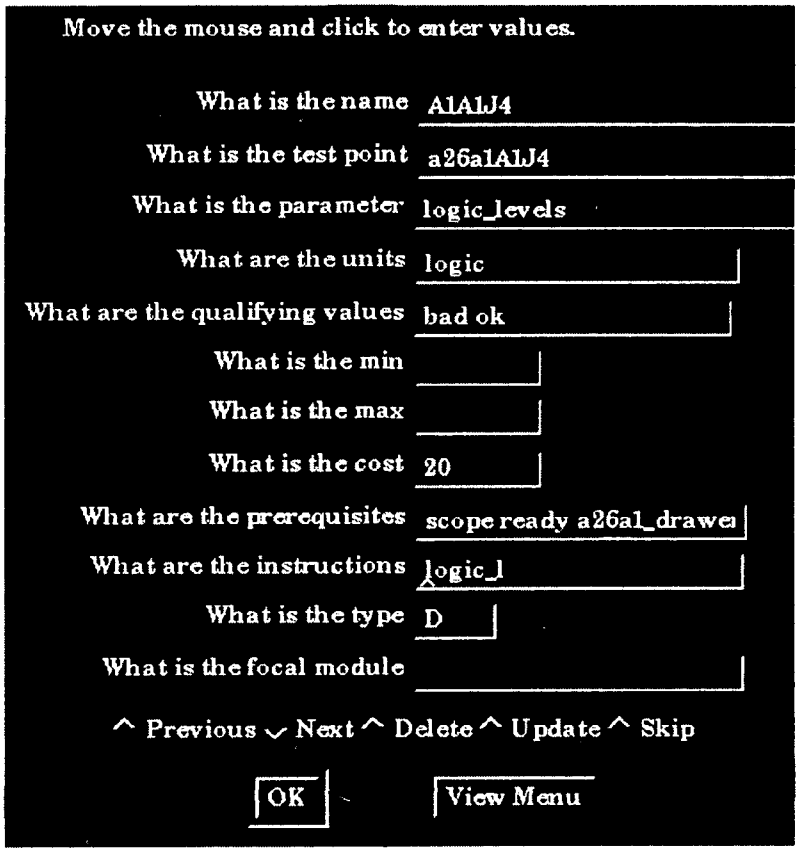


Fig. 8 - *View or Modify Data Records* window for a test database

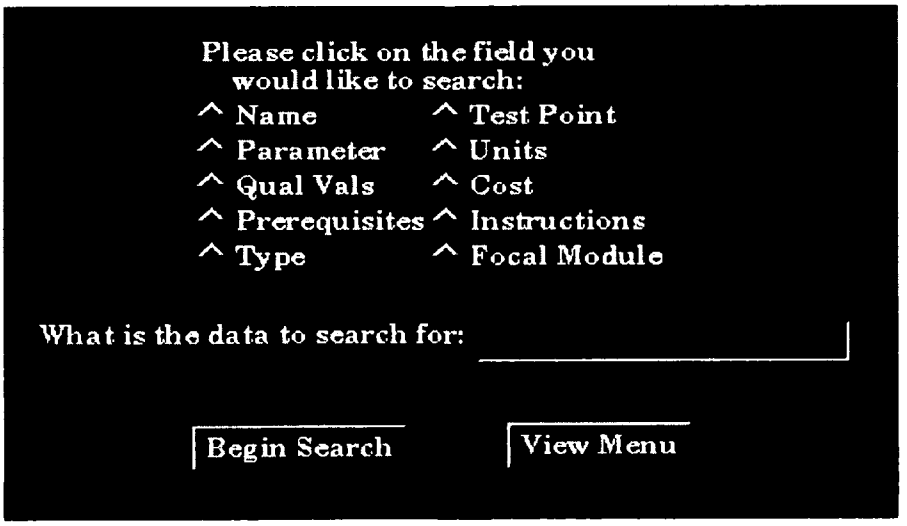


Fig. 9 - Search window for a test database search

button (on a 2-button mouse, both together) to copy data. This can cut down on repetitious typing while entering similar records.

After the record is completed, there are three remaining options on what to do: Add it to the database and add another, add it to the database and return to the Main Menu, or do not add it but return to the Main

Menu. The Main Menu here refers to the calling menu; it may be the View Menu or the Data Manipulation Menu.

CONCLUSION

Through the use of X-Windows, a highly portable, robust GUI was developed. Full Empress functionality was maintained with increased ease of use over the text based system FISDM [3]. There are two major enhancements over the previous work. The creation of the knowledge database can now be performed primarily by graphical interactions with the interface. The interface also facilitates the textual entry of data records by providing templates, for rule and test databases, with point and click type editing features. This allows the tool to have the character of typical design tools used in other fields of science and engineering. In addition to the graphical interface, an expert system using the CLIPS shell is embedded in the FISDMPro tool. The expert system contains knowledge of the rules required for knowledge database assembly, and automates much of the effort required to enter the rule and test information, as well as integrating the process into the single operation of drawing a model.

A benefit of embedding the expert system for assembly of the knowledge base is that although the FISDMPro interface is tailored for the FIS format, it would require only modification of the rules in the expert system portion to handle other knowledge base formats. Of course, to operate properly with the rest of the system, modifications to other software components (database interface and conversion routines) would be required. This is merely intended to emphasize the flexibility of this approach for knowledge acquisition.

Future goals would include the possibility of scanning schematic diagrams into the computer, in effect fully automating the task. These scanned diagrams would place modules and their connections in the proper places. An additional extension would be to capture the desired information for knowledge base creation during the design phase of any system that would use FIS for fault diagnosis.

REFERENCES

1. F. Pipitone, K.A. DeJong, W. Spear, "An Artificial Intelligence Approach to Analog Systems Diagnosis," NRL Report 9219, Sept. 1989.

2. J. Molnar, "Creation, Validation, Testing and Data Management of a Knowledge Base Designed for a Technician's Assister System for the AN/SQS-53B, Unit 26, Using a Fault Isolation System Shell," NRL Report 9296, Dec., 1990.

3. C.B. Barclay and J. A. Molnar, "Development of a Fault Isolation System Database Manager," NRL Memorandum Report 92-6944, Feb., 1992.

4. Eisenstadt, Domingue, Rajan and Motta, "Visual Knowledge Engineering," *IEEE Transactions on Software Engineering*, **16**(10), (1990).

5. J. C. Giarratano, "Clips Users Manual." CLIPS version 5.1, NASA, Lyndon B. Johnson Space Center, Information Systems Directorate, Software Technology Branch, Sept. 10, 1991.

6. Swaby, "VIDES: An Expert System for Visually Identifying Microfossils," *IEEE Expert*, (April 1992).

Appendix

TUTORIAL EXAMPLE

This Appendix is provided to demonstrate the basic concepts of using FISDMPro to create a knowledge base structure that is acceptable to FIS. This is intended only to demonstrate FISDMPro and not discuss knowledge base creation in general. For a detailed description of the FIS knowledge base structure and components the reader should refer to references 1 through 3.

For this example the sample Unit Under Test (UUT) consists of four replaceable modules that are connected. Figure A1 shows a diagram of this UUT. In this example the replaceable modules are the items represented by the boxes. The connections between modules are represented by lines with arrowheads. The direction of the arrow indicates the direction of flow for items passed between modules. From this flow causal relationships are formed. In its simplest form an incorrect output from a module can be caused by a fault in the module or by a faulty input to the module. The latter relationship depends on the internal connection of input to outputs within the module. In the example the internal connections are indicated by the dashed lines. The object of FISDMPro is to permit the creation of a FIS knowledge base by drawing in the relationship and providing information pertinent to the physical relationship of the UUT.

The basic process consists of: (a) defining the modules, (b) defining the connections between modules, (c) defining the internal connections of the module and (d) generating a FIS knowledge base.

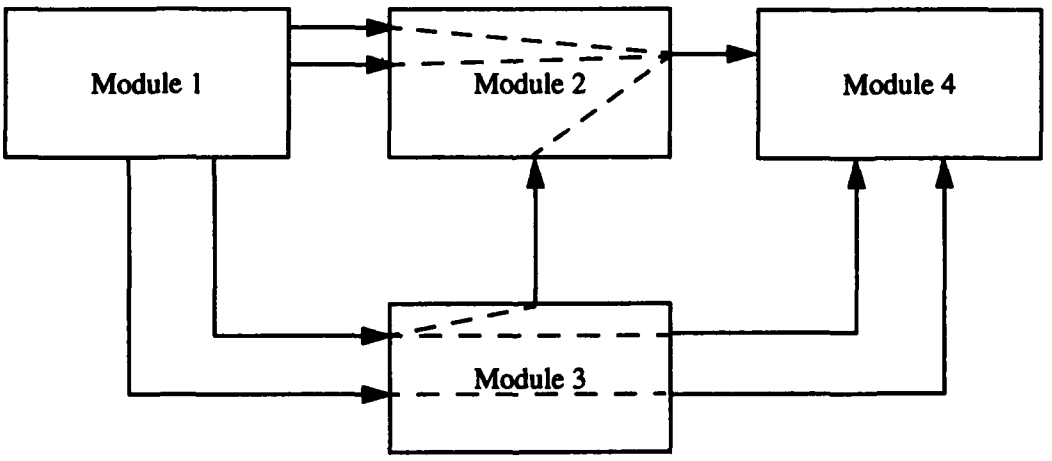


Fig. A1 - Example of a simple Unit Under Test to demonstrate FISDMPro capabilities

The first step in the process is to define the modules. Once FISDMPro is started a window is presented that consists of two areas: a command area and a drawing area. Figure A2 shows this window. The command area is located on the left edge of the window. It consists of an area to enter module naming information at the top left, a list of commands below the name entry slot, a quit button to exit the program, and a locator entry area to define locations of modules. The drawing area is the large area to the right of the command area. It has a scroll bar at the bottom to allow creation and viewing of UUTs that can not comfortably fit in the basic viewable drawing area. In Fig. A2 the command to add a module, *Add Module, has* been selected. The user may then enter a name for this module in the name entry slot. Naming modules is not required but very desirable. Then to define the module, the user simply places the mouse in the drawing area and depresses the right mouse button. With the mouse button depressed, the mouse is dragged toward the bottom right. As this is done, a rectangle is created on the screen. Releasing the mouse button then defines the size of the module. The name of the module is automatically placed in the top left corner of the newly defined module. In Fig. A2 module 1 is defined and given the name "mod1".

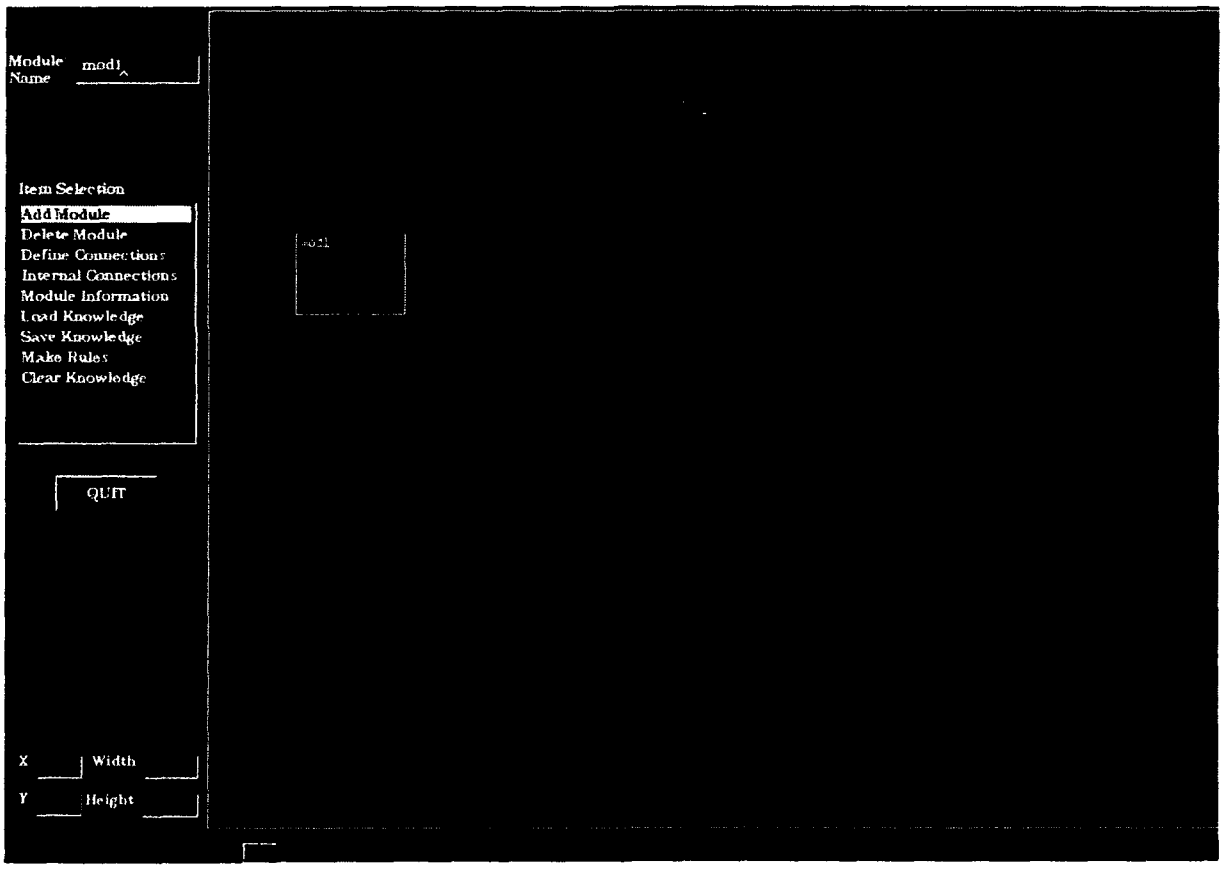


Fig. A2 - FISDMPro window during addition of first module

The *Add Module* command remains active until another command is selected. This makes the creation of modules the most expeditious command to perform. Only the name needs to be provided and the next module can be added. This imposes a methodology, of sorts, since even though other commands can be easily selected, it becomes easier to define all of the modules first. This feature is meant to provide some intuition to the user, since other commands such as *Defining Connections* and *Internal Connections*

rely on the fact that more than one module should exist to allow connections to be defined. With the definition of modules being the preferred command, the rest of the UUT is defined, as seen in Fig. A3.

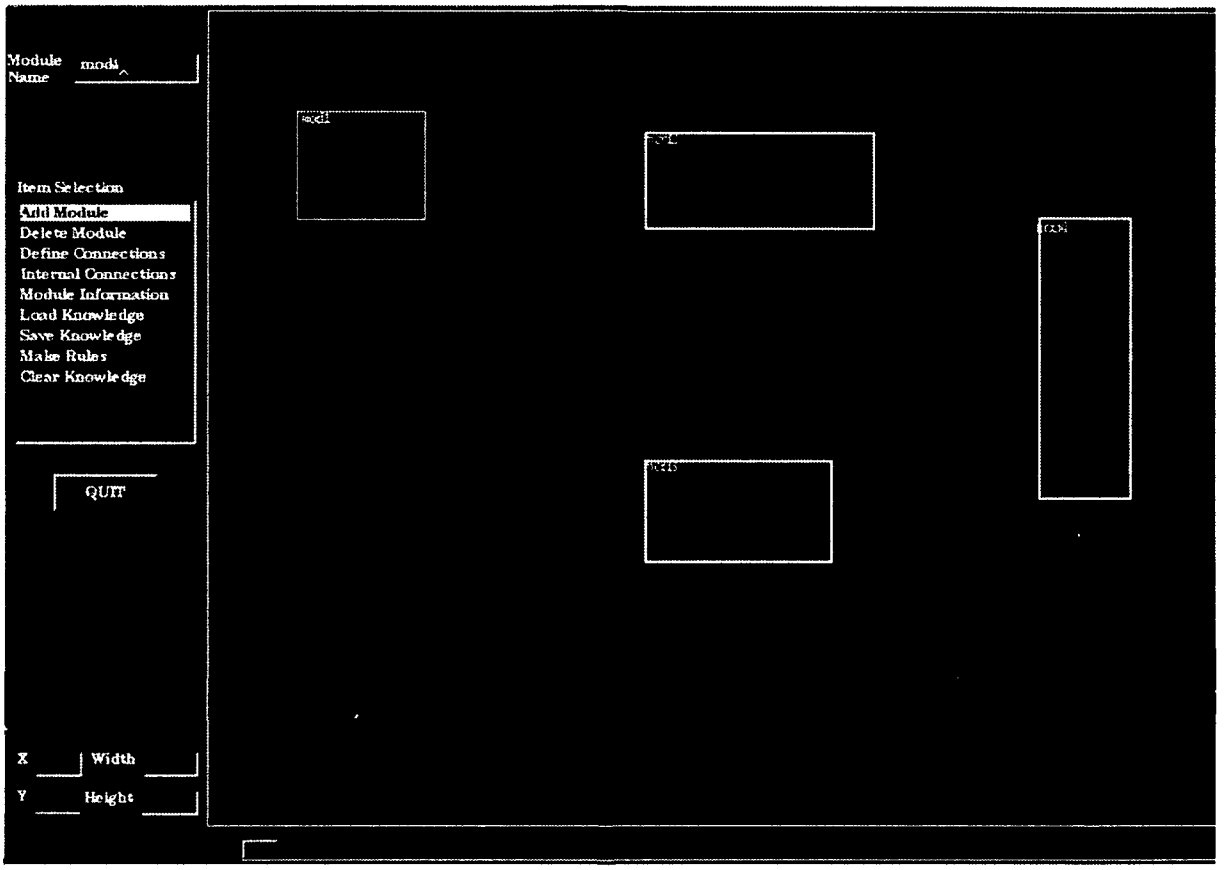


Fig. A3 - FISDMPro window after all modules added

The next step is to define the connections between modules. This is performed by selecting the *Define Connections* command from the command list. Once the command is selected, the user must go to the drawing portion of the window and define which module is providing the output. This is done by clicking once on the appropriate module. Next, the module that contains the input connection is selected by clicking with the mouse on a second module. When these actions are performed, the module where the output connection exists is highlighted green. The module that contains the input connection is highlighted red. The intuition to this color selection is green for start (or go) and red for stop. A window is displayed and the user is asked to fill in information on this connection. Generally, the link name should be the name of either the input terminal on the module highlighted in red, or the terminal name of the output terminal on the module highlighted in green. Either one is acceptable. However, it is useful for the user to develop a convention of his own and be consistent in using it. In this example, a link name of link3-2-1 is used with the convention being the link from module 3 to module 2, and this being the first link defined between module 3 and module 2. The condition that is intended for this example is bad, and the parameter is provided a value of volts. This connection is intended to be testable, so the Testpoint check box is checked. The user provides the values for the Units, Min and Max boxes. Doing this allows FISDMPro, during the rule generation step, to also generate test information for the FIS knowledge base. Figure A4 shows the

process of defining the connection. Notice that other connections have already been defined and the definition is noted on the drawing by placing lines between the modules that are connected. Each of these connections are simple connections. This means that they are not used as testpoints and no testpoint information exists in those connections. A testpoint is designated with a small white dot beside the line connection, as Fig. A5 shows.

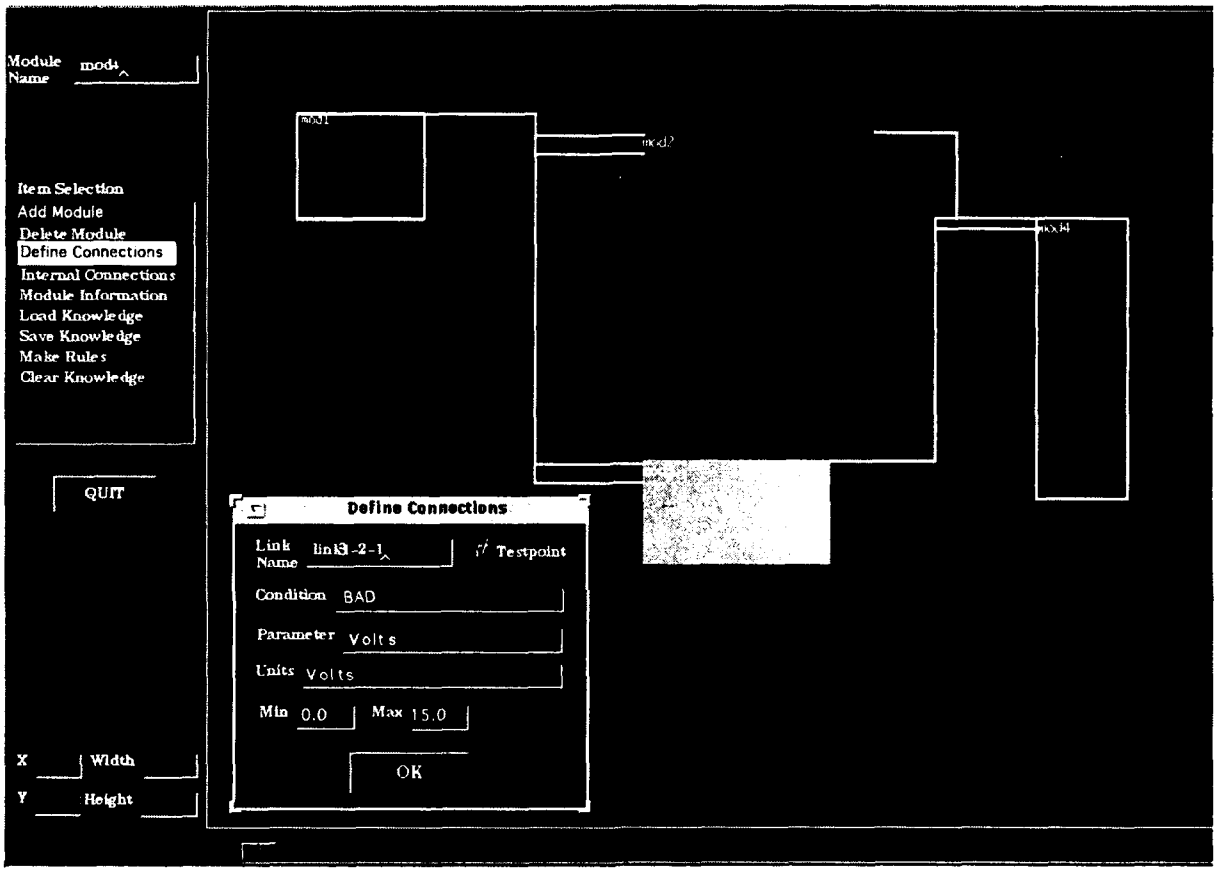


Fig. A4 - FISDMPro window during connection definition

At this point in the process all of the connections between modules have been made; however, the connections internal to the module that relate the module inputs and to the module outputs have not been defined. This is achieved by selecting the *Internal Connections* command by placing the mouse over the command and clicking. After selecting this command, a module for which the internal connections are to be defined must be selected. This is done by placing the mouse over a module and clicking. Once the module is selected, a window as the one shown in Fig. A6 is presented. The window shown in Fig. A6 is for module 2. The first column on the window, called "Incoming Connections," displays a list of all connections that act as inputs to the module; in this case there are three inputs. The next column, called "Outgoing Connections," lists all connections that are outputs from the module. To define an internal connection, the user selects one connection from the "Incoming Connection" list and one connection from the "Outgoing Connection" list. By doing this the connections that are selected are placed in the column labeled "Selections." After selecting the connections, the OK button is selected to define the connection. This process is repeated until all the internal connections are defined. Any input connections that are not

defined as being connected to an output are, by default, connected to all output connections. Similarly, if an output connection is not defined, the FISDMPro by default defines the internal connections as all input connections being connected to the output connection. If the user does not wish to define any internal connections, then FISDMPro by default will provide internal connections for all modules by connecting every input of each module to every output of the module. The reason for this is that internal connections must be defined in order to generate a proper FIS knowledge base. If connections are not defined, FIS warns the user of the situation during the compilation process.

Once the Internal Connections are defined, or during the process of defining the connections, the connections can be examined by selecting the *Module Information* command. This command presents a window similar to that in Fig. A6, but has Next and Previous buttons to review the connections. The connections for all modules in the unit can be reviewed in this manner. When the user is satisfied with the definition of the modules and their interconnections, the user can select the *Make Rules* command to generate the FIS knowledge base.

The FIS knowledge base is created by selecting the *Make Rules* command from the command list. When this command is selected, FISDMPro processes the information that has been provided and creates a workable knowledge base. In general this takes some time, but the result is a file that has the contents similar to that in Fig. A7, for the example module. The two major components of the FIS knowledge base are the rule and test information; FISDMPro automatically creates these portions in a format ready for compilation with FIS.

This concludes the tutorial portion of the report. Other features are described in the main text of this report. Information regarding FIS knowledge base requirements can be found in Refs. 1 and 2, while information on the database and format conversion procedures are described in Ref. 3.

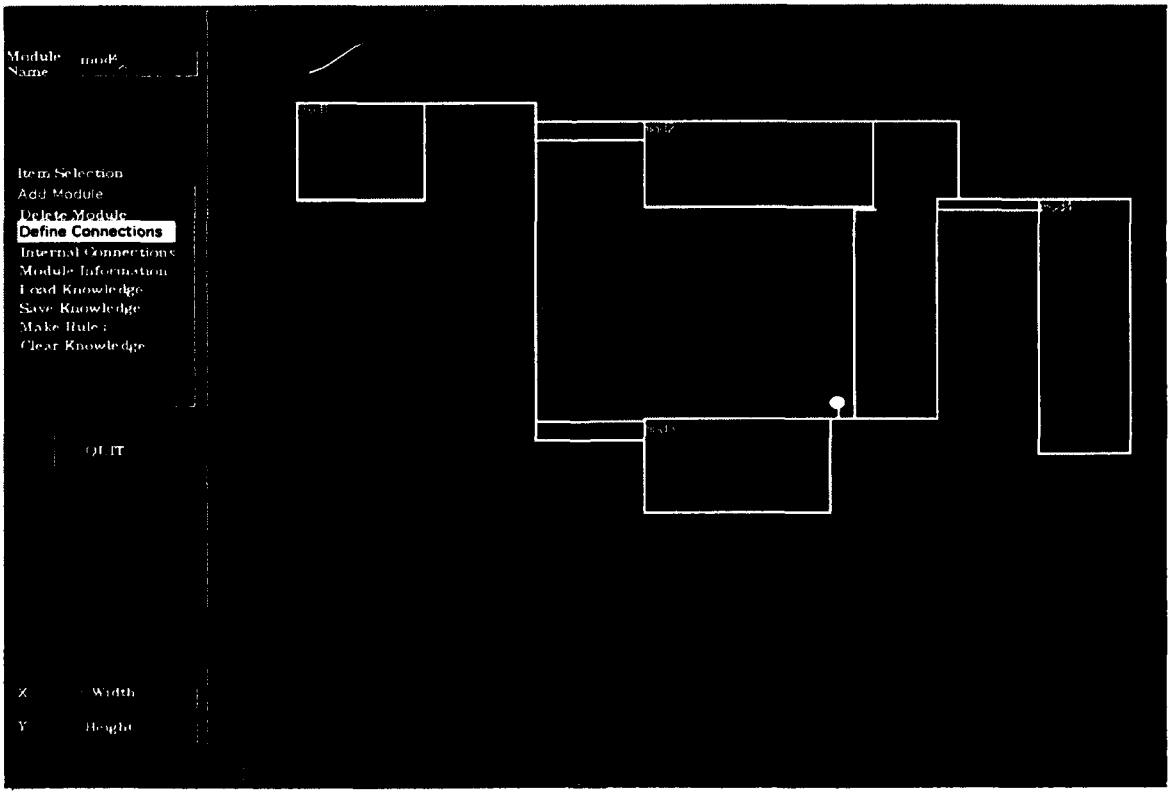


Fig. A5 - FISDMPro window indicating connections and testpoint locations

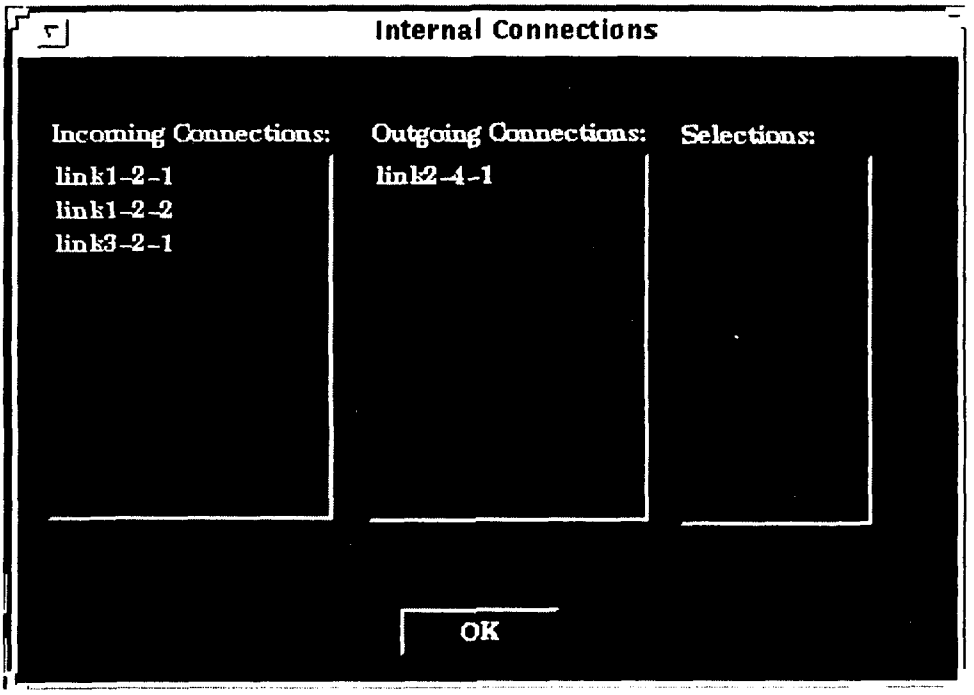


Fig. A6 - FISDMPro window during connection definition

```
NIL
NIL
NIL
(
((NAME mod1) (FRATE 0.25)
  (CAUSAL-RULES
   (
    (T mod1
     (link1-3-2 volts BAD))
    (T mod1
     (link1-3-1 freq BAD))
    (T mod1
     (link1-2-1 volts BAD))
    (T mod1
     (link1-2-2 freq HI))
   )))
((NAME mod2) (FRATE 0.25)
  (CAUSAL-RULES
   (
    (T mod2
     (link2-4-1 volts BAD))
    (T (link3-2-1 volts BAD)
     (link2-4-1 volts BAD))
    (T (link1-2-1 volts BAD)
     (link2-4-1 volts BAD))
    (T (link1-2-2 freq HI)
     (link2-4-1 volts BAD))
   )))
((NAME mod3) (FRATE 0.25)
  (CAUSAL-RULES
   (
    (T mod3
     (link3-4-1 amp BAD))
    (T mod3
     (link3-4-2 freq BAD))
    (T mod3
     (link3-2-1 volts BAD))
    (T (link1-3-2 volts BAD)
     (link3-4-2 freq BAD))
    (T (link1-3-2 volts BAD)
     (link3-4-1 amp BAD))
    (T (link1-3-1 freq BAD)
     (link3-2-1 volts BAD))
   )))
)
((link3-2-1(link3-2-1 volts
     SI
     ((OK(0.0 15.0)
     (BAD((-inf 0.0)(15.0 inf))))
     VOLTS
     PERF
     1.0
     NIL)
)
NIL
NIL
NIL
```

Fig. A7 - An uncompiled FIS knowledge base generated for the example with FISDMPro